CLAIMS

What is claimed is:

1. A method of emulating an emulated transaction processing system to enable compatibility testing of said emulated system with a testing system, comprising the steps of:

defining a test plan;

identifying unique aspects of the tested system;

modifying scripts to model said emulated system, including said unique aspects; and

permitting said testing system access to a server running said scripts.

2. A method as in claim 1, wherein said emulated system uses a messaging engine.

3. A method as in claim 2, wherein said messaging engine is a FIX engine.

4. A method as in claim 1, wherein said scripts comprise session-level scripts.

5. A method as in claim 1, wherein said scripts comprise application-level scripts.

6. A method as in claim 2, wherein said step of modifying scripts comprises the steps of:

dynamically determining the behavior of the messaging engine by running tests and logging test results;

parsing said logged test results; and

determining how said messaging engine validates incoming messages and composes outgoing messages.

7. A method as in claim 6, wherein tests are run, logged, and parsed using test scripts.

8. A method as in claim 6, further comprising adding customized wizpages.

9. A method of testing compatibility of a testing system and an emulated transaction processing system, comprising the steps of:

establishing communication with an emulating system that emulates behavior of the emulated system using scripts;

receiving a test data request from said emulating system;

sending test data to said emulating system, wherein said test data corresponds to said test data request; and

receiving a message from said emulating system indicating that said test data was successfully received.

10. A method as in claim 9, wherein said test data comprises an order.

11. A method as in claim 9, further comprising the step of providing a visual indication of test results to a user.

12. A method of emulating an emulated transaction processing system to enable compatibility testing of said emulated system with a testing system, comprising the steps of:

emulating behavior of said emulated system using scripts;

establishing communication with the testing system;

sending a test data request to said testing system;

receiving test data from said testing system, wherein said test data corresponds to said test data request; and

sending a message to said testing system indicating that said test data was successfully received.

13. A method as in claim 12, wherein said test data comprises an order.

14. A method as in claim 12, further comprising the step of providing a visual indication of test results to a user.

15. Software for emulating an emulated transaction processing system to enable compatibility testing of said emulated system with a testing system, comprising:

user interface software;

script writing software;

application messaging software using a network connection;

script executing software; and

test scripts.

16. Software as in claim 15, wherein said messaging software uses a FIX protocol.

17. Software as in claim 15, wherein said messaging software uses a FIXML protocol.

18. Software as in claim 15, wherein said messaging software uses a FPML protocol.

19. Software as in claim 15, wherein said messaging software uses a SWIFT protocol.

20. Software as in claim 15, wherein one or more of said test scripts comprise event handling script functions.

21. Software as in claim 15, further comprising dynamic HTML generating software controllable by a script.

22. Software as in claim 15, further comprising software configured to provide a graphical user interface.

23. Software as in claim 22, wherein said graphical user interface is enabled to provide a visual indication of test results.

24. Software for emulating an emulated transaction processing system to enable compatibility testing of said emulated system with a testing system, comprising:

a protocol definition;

a first set of scripts implementing said protocol definition; and

a second set of scripts linked to the first set of scripts, said second set of scripts emulating a transaction processing system that implements said protocol definition.

25. Software as in claim 24, comprising:

a script interpreting engine embedded into a messaging engine; and

an application programming interface between said messaging engine and said second set of scripts, wherein said application programming interface is configured to pass inbound and outbound messages to said second set of scripts.

26. Software as in claim 25, wherein said application programming interface is configured provide to said second set of scripts access to attributes of said messages.

27. Software as in claim 25, wherein said application programming interface is configured to enable said second set of scripts to handle inbound and outbound messaging errors.

28. Software as in claim 25, wherein said application programming interface is configured to enable said second set of scripts to control messaging engine behavior by returning true or false values from script functions.

29. Software as in claim 25, wherein said application programming interface is configured to provide said second set of scripts with access to services provided by a protocol engine or an application connected thereto.

30. Software as in claim 25, further comprising software for associating one or more scripts with one or more event sources, such that an instance of a script maintains program state across several different events and messages.

31. Software as in claim 25, further comprising software for declaring scripts and enabling said scripts to include libraries of other scripts.